HAL
archives-ouvertes.fr

Evaluation de polynômes et de fractions rationnelles sur FPGA avec des opérateurs à additions et décalages en grande base.

Romain Michard, Arnaud Tisserand

▶ **To cite this version:**

Romain Michard, Arnaud Tisserand. Evaluation de polynômes et de fractions rationnelles sur FPGA avec des opérateurs à additions et décalages en grande base.. [Research Report] Laboratoire de l'informatique du parallélisme. 2004, 2+10p. hal-02102022

HAL Id: hal-02102022

https://hal-lara.archives-ouvertes.fr/hal-02102022

Submitted on 17 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

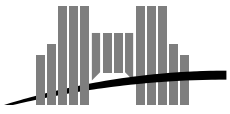
Evaluation de polynômes et de fractions rationnelles sur FPGA avec des opérateurs à additions et décalages en grande base

Romain Michard, Arnaud Tisserand et Nicolas Veyrat-Charvillon

Décembre 2004

Rapport de recherche N° 2004-62

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`



Evaluation de polynômes et de fractions rationnelles sur FPGA avec des opérateurs à additions et décalages en grande base

Romain Michard, Arnaud Tisserand et Nicolas Veyrat-Charvillon

Décembre 2004

Abstract

This work deals with FPGA arithmetic operators, based on shift-and-add algorithm, for polynomial and rational approximation of functions. These operators are high-radix iterations of the E-method proposed by M. Ercegovac. Our results show high performances by mixing the simple architecture of shift-and-add algorithms and the generic nature of polynomial and rational approximations.

Keywords: computer arithmetic, hardware arithmetic operator, polynomial evaluation, rational fraction evaluation, E-method.

Résumé

Ce travail porte sur l'étude et l'implantation FPGA d'opérateurs arithmétiques, à base d'additions et de décalages, pour l'approximation polynomiale et rationnelle de fonctions. Ces opérateurs sont des versions en grande base de l'itération de E-méthode proposée par M. Ercegovac. Les résultats montrent que ces opérateurs présentent de bonnes performances en alliant la simplicité des architectures à base d'additions et de décalages et le caractère générique des approximations polynomiales et rationnelles.

Mots-clés: arithmétique des ordinateurs, opérateur arithmétique matériel, évaluation de polynôme, évaluation de fraction rationnelle, E-méthode.

1 Introduction

L'implantation matérielle d'opérateurs arithmétiques évolués est un besoin important dans bon nombre d'architectures spécialisées en traitement du signal et des images ou en contrôle numérique. Dans ce papier, nous étudions des opérateurs arithmétiques pour l'évaluation de fonctions algébriques (division, racine carrée...) et de fonctions élémentaires (sinus, cosinus, exponentielle, logarithme, arc-tangente...).

Trois grandes classes d'algorithmes sont utilisées pour l'évaluation des fonctions élémentaires [12] : les algorithmes à base d'approximations polynomiales ou rationnelles, les algorithmes à base de tables et enfin les algorithmes à base d'additions et de décalages.

Les approximations polynomiales ou rationnelles sont essentiellement utilisées en logiciel. Les polynômes sont évalués en utilisant le schéma de Horner. Il est assez simple d'obtenir un polynôme d'approximation d'une fonction $f(x)$ en utilisant l'algorithme de Remes [14]. Des approximations polynomiales et rationnelles des principales fonctions utilisées en calcul scientifique peuvent être trouvées dans [9]. Lorsqu'elles sont utilisées en matériel, on essaye souvent d'utiliser les caractéristiques de coefficients particuliers pour réduire la surface des multiplieurs (voir par exemple [13, 7]). Le principal problème pour l'évaluation des fractions rationnelles est le coût de la division en temps et en surface de circuit. Ces limitations font que seules des solutions à base d'arithmétique sérielle [10] ou de fractions continues [17] sont proposées pour l'approximation rationnelle en matériel.

Les méthodes à base de tables reposent sur l'utilisation de petites tables et d'un petit nombre d'opérations très simples comme des additions. Ces méthodes sont limitées aux petites précisions, jusqu'à une vingtaine de bits [15, 3]. De plus, ces méthodes sont spécifiques à chaque fonction évaluée. L'implantation d'opérateurs permettant d'évaluer plusieurs fonctions n'est donc pas envisageable avec ces méthodes.

Enfin, la classe des algorithmes à base d'additions et de décalages qui fournissent un chiffre du résultat à chaque cycle de calcul. Dans cette classe, on trouve SRT [5] pour la division et la racine carrée ou CORDIC [16, 18] pour certaines fonctions élémentaires. Ces méthodes permettent de réaliser des opérateurs de taille modérée et présentent une architecture simple. Toutefois, ces méthodes sont spécifiques à un petit nombre de fonctions et ne permettent donc pas d'avoir un opérateur générique. La E-méthode, proposée par M. Ercegovac [4, 6], permet d'évaluer des polynômes et des fractions rationnelles avec une itération à base d'additions et de décalages proche de celle utilisée pour la division et la racine carrée. La E-méthode avait été utilisée pour de l'évaluation de polynômes en arithmétique en-ligne (en série avec les poids forts en tête) dans [8] et en base 2.

Nous présentons dans ce travail une implantation sur circuit FPGA d'opérateurs de E-méthode pour des approximations polynomiales et rationnelles. Les itérations de la E-méthode ne nécessitant que des additions et des décalages, aucun multiplieur ou diviseur n'est nécessaire dans le circuit final. Le but est ici d'allier le caractère générique des approximations polynomiales ou rationnelles et la simplicité des architectures à base d'additions et de décalages. Les opérateurs obtenus sont facilement réutilisables et fonctionnent en arithmétique parallèle sur des surfaces de circuit modérées.

La section 2 présente quelques rappels sur la E-méthode et les approximations polynomiales et rationnelles en grande base ($\beta \in \{2, 4, 8\}$). L'étude et l'implantation de la E-méthode sur circuit FPGA sont présentées dans la section 3. Nous comparons nos implantations avec d'autres travaux à la section 4. Enfin, nous concluons et donnons quelques perspectives en section 5.

2 Rappels sur la E-méthode

La méthode d'évaluation, ou E-méthode, a été proposée par M. Ercegovac dans les années 70 [4, 6]. Cette méthode permet de résoudre certains systèmes linéaires, à diagonale dominante, à l'aide d'une itération simple et régulière à base d'additions et de décalages. Les systèmes linéaires cibles sont de la forme :

$$\begin{pmatrix} 1 & -x & 0 & & \cdots & & & 0 \\ q_1 & 1 & -x & 0 & & \cdots & & 0 \\ q_2 & 0 & 1 & -x & 0 & & \cdots & 0 \\ & & \ddots & \ddots & \ddots & \ddots & & \vdots \\ & & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & \ddots & \ddots & \ddots & 0 \\ & & & & & & 1 & -x \\ q_n & & & \cdots & & & 0 & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ \vdots \\ \vdots \\ p_{n-1} \\ p_n \end{pmatrix} \quad (1)$$

On note $\mathcal{A}$ la matrice de ce système, b le vecteur second membre et y le vecteur solution. La taille du système est $n+1$.

Après résolution du système 1, la première composante du vecteur solution y est la valeur au point x de la fraction rationnelle R, de degré n, dont les coefficients du numérateur sont les composantes de b et ceux du dénominateur sont les composantes de la première colonne de $\mathcal{A}$. C'est à dire que la solution de $\mathcal{A}y = b$ est $y = [y_0, y_1, \ldots, y_n]^t$ telle que

$$y_0 = R(x) = \frac{P(x)}{Q(x)} = \frac{p_n x^n + p_{n-1} x^{n-1} + \cdots + p_0}{q_n x^n + q_{n-1} x^{n-1} + \cdots + 1}.$$

La E-méthode permet donc d'évaluer des fractions rationnelles en un point. En toute généralité, les degrés des polynômes au numérateur et au dénominateur de $R(x)$ peuvent être différents (n est alors le plus grand). En pratique, il semble que d'un point de vue de la précision des approximations réalisées, le cas des degrés égaux (ou très proches) soit à privilégier.

La formulation du système linéaire 1 pouvant être résolu par la E-méthode impose que le polynôme au dénominateur $Q(x)$ soit tel que $q_0 = 1$. En pratique, cette limitation n'est pas problématique. En effet, il existe des techniques de mise à l'échelle permettant de modifier une fraction rationnelle en une fraction rationnelle dont les coefficients respectent cette contrainte.

On trouve dans [4, 6] les limites sur les valeurs possibles pour les coefficients de la fraction rationnelle et sur l'argument x. Des techniques de mise à l'échelle permettent de limiter l'impact de ces contraintes. En règle générale, l'évaluation d'une fonction élémentaire se fait en deux étapes : la réduction d'argument puis l'évaluation proprement dite [12]. La phase de réduction d'argument permet de se ramener à un petit domaine dans lequel l'approximation utilisée est suffisamment précise.

Dans [2], on trouve une méthode permettant de savoir si une fraction rationnelle est calculable à l'aide de la E-méthode. La bibliothèque MEPLib [1] devrait être capable dans un futur proche de fournir des polynômes et des fractions rationnelles avec des contraintes sur leurs coefficients et répondant aux exigences des algorithmes de réduction d'argument classiques.

En simplifiant légèrement la matrice $\mathcal{A}$, la E-méthode permet aussi d'évaluer des polynômes. En effet, si tous les q_i sont nuls (sauf $q_0 = 1$), alors la première composante du vecteur solution est la valeur au point x du polynôme P, de degré n, dont les coefficients sont les composantes de b. C'est à dire, le polynôme est $Q(x) = 1$ et la solution de $\mathcal{A}y = b$ est alors $y = [y_0, y_1, \ldots, y_n]^t$ avec

$$y_0 = P(x) = p_n x^n + p_{n-1} x^{n-1} + \cdots + p_0.$$

Avant de présenter l'itération de la E-méthode, nous devons introduire quelques notations utiles pour la suite. La base du système de représentation des nombres est notée β (en pratique, $\beta \in \{2, 4, 8\}$ dans ce travail). Le vecteur des restes partiels, de taille $n+1$, est noté w. Les différentes valeurs d'une quantité dans le temps sont représentées avec la notation crochet (comme en traitement du signal). Par exemple $w[j]$ dénote le vecteur des restes partiels à la $j$ème itération. Le vecteur de chiffres du résultat trouvé à chaque itération j est noté $d[j]$.

L'algorithme de E-méthode est présenté en figure 1. Le vecteur des restes partiels est initialisé avec les coefficients du polynôme P (les composantes de b). Le premier vecteur de chiffres du résultat est le vecteur nul.

1 *initialisation :*
2 $w[0] \longleftarrow b$
3 $d[0] \longleftarrow 0$
4 *itération :*
5 **pour** j **de** 1 à m **faire**
6 $w[j] \longleftarrow \beta \times \Big(w[j-1] - \mathcal{A} \times d[j-1] \Big)$
7 $d[j] \longleftarrow S(w[j])$
8 *résultat :*
9 $y_0[m] = \sum_{i=1}^{m} d_0[i]\beta^{-i}$

FIG. 1 – Algorithme d'évaluation avec la E-méthode (version vectorielle).

Comme tous les algorithmes à base d'addition et de décalages, la E-méthode produit un chiffre du résultat à chaque itération en commençant par les poids forts. La concaténation des différents chiffres fournit une valeur qui tend vers la valeur mathématique du résultat (à l'infini). Ici, le résultat de chaque itération est un vecteur de chiffres $d[j]$. L'itération est basée sur un calcul similaire à celui d'une division où l'on "diviserait" par la matrice $\mathcal{A}$ (d'où le terme reste partiel pour w). Pour chaque ligne $i = 1, \ldots, n-1$ de la matrice $\mathcal{A}$, le calcul effectué est :

$$w_i[j] = 2 \times \big(w_i[j-1] - d_0[j-1]q_i - d_i[j-1] + d_{i+1}[j-1]x \big) \tag{2}$$

Dans les cas $i = 0$ et $i = n$, le calcul se simplifie en $w_0[j] = 2 \times \big(w_0[j-1] - d_0[j-1] + d_1[j-1]x \big)$ et $w_n[j] = 2 \times \big(w_n[j-1] - d_0[j-1]q_n - d_n[j-1] \big)$.

Le calcul des nouveaux termes du reste partiel n'implique que des additions/soustractions et des produits d'un nombre par un seul chiffre. On verra en section 3 ce qu'il se passe lorsque β augmente.

A chaque itération, un nouveau vecteur de chiffres du résultat $d[j]$ est produit. Ce calcul se fait en utilisant la fonction de sélection S définie dans un cadre général par l'expression 3. Nous présentons en section 3 les détails de la fonction de sélection pour les différentes bases utilisées dans ce travail.

$$S(x) = \begin{cases} \text{signe } x \times \lfloor |x + 1/2| \rfloor, & \text{si } |x| \leq 1 \\ \text{signe } x \times \lfloor |x| \rfloor, & sinon, \end{cases} \tag{3}$$

3 Etude et implantation FPGA de la E-méthode

3.1 Architecture des opérateurs

Le calcul de l'itération présentée dans l'algorithme 1 est découpé suivant les lignes de la matrice $\mathcal{A}$. Le calcul correspondant à chaque ligne de l'itération, équation 2, est confié à une unité de calcul. L'architecture générale de ces unités est représentée en figure 2.

Le calcul effectué dans chaque unité est assez simple. Il se limite à 2 ou 3 additions/soustractions, des multiplications d'un chiffre par un nombre et la fonction de sélection S. Comme dans la suite nous visons des implantations sur circuits FPGA, nous choisissons de représenter le reste partiel w_i en complément à deux pour bénéficier des lignes d'addition rapide présentes dans les FPGA.

Les produits d'un chiffre par un nombre peuvent être particulièrement simples suivant la base utilisée. Dans le cas de la base $\beta = 2$, l'ensemble de chiffres utilisé est $\mathcal{E}_{2,1} = \{-1, 0, 1\}$, le produit se résume alors à un simple multiplexeur et à l'utilisation d'un additionneur/soustracteur. Dans le cas de la base $\beta = 4$, deux ensembles de chiffres sont possibles $\mathcal{E}_{4,2} = \{-2, -1, 0, 1, 2\}$ et $\mathcal{E}_{4,3} = \{-3, -2, -1, 0, 1, 2, 3\}$. Le premier nécessite un multiplexeur plus grand et un décalage constant (routage). Dans le cas du second, il faut aussi former le terme $3x$ ou $3q_i$ en utilisant un additionneur ($3x = 2x + x$). On remarque que la formation de ces produits ne se fait qu'une seule fois en début de calcul pour $3x$ et une seule fois à la configuration de Q dans l'unité pour $3q_i$. Le calcul de ces petits produits n'est donc pas un problème.

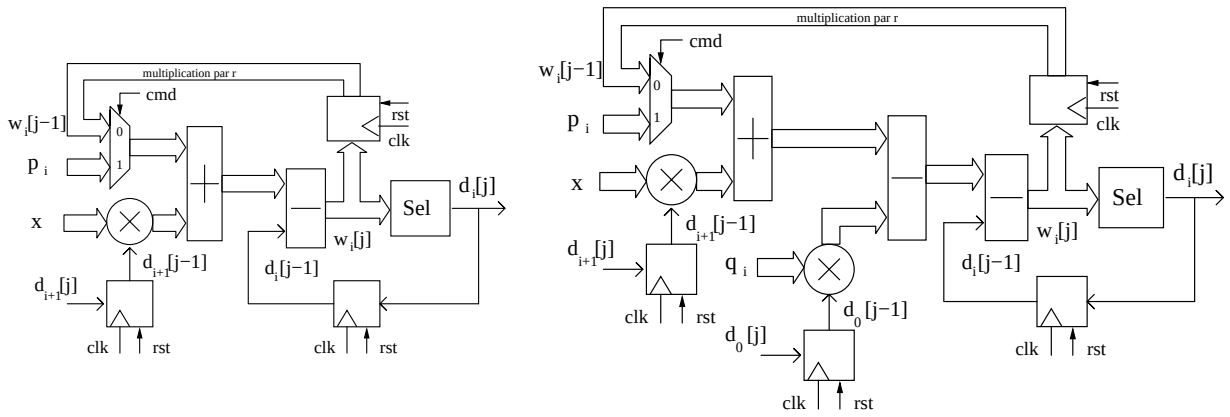


FIG. 2 – Unité de calcul d'une ligne de l'itération à l'étape j dans le cas polynomial (à gauche) et dans le cas rationnel (à droite).

Le calcul effectué par la fonction de sélection S est représenté dans la table 1 pour les bases 2 et 4 (avec les deux ensembles de chiffres possibles $\mathcal{E}_{4,2}$ et $\mathcal{E}_{4,3}$ pour $\beta = 4$). Le calcul pour la base 8 est similaire mais sa présentation complète nécessite trop de place.

	$\beta = 2, \mathcal{E}_{2,1}$	
$\hat{w}[j-1]$	$d[j]$	$w[j] \in$
0000	0	$[0, 1/2[$
0001	1	$[1/2, 1[$
0010	1	$[1, 3/2[$
0011	1	$[3/2, 2[$
0100	n/a	impossible
⋮		
1011	n/a	impossible
1100	-1	$[-2, -3/2[$
1101	-1	$[-3/2, -1[$
1110	-1	$[-1, -1/2[$
1111	0	$[-1/2, 0[$

	$\beta = 4, \mathcal{E}_{4,2}$		$\beta = 4, \mathcal{E}_{4,3}$	
$\hat{w}[j-1]$	$d[j]$	$w[j] \in$	$d[j]$	$w[j] \in$
00000	0	$[0, 1/2[$	0	$[0, 1/2[$
00001	1	$[1/2, 1[$	1	$[1/2, 1[$
00010	1	$[1, 3/2[$	1	$[1, 3/2[$
00011	2	$[3/2, 2[$	2	$[3/2, 2[$
00100	2	$[2, 5/2[$	2	$[2, 5/2[$
00101	2	$[5/2, 3[$	3	$[5/2, 3[$
00110	n/a	impossible	3	$[3, 7/2[$
00111	n/a	impossible	3	$[7/2, 4[$
01000	n/a	impossible	n/a	impossible
⋮				
10111	n/a	impossible	n/a	impossible
11000	n/a	impossible	-3	$[-4, -7/2[$
11001	n/a	impossible	-3	$[-7/2, -3[$
11010	-2	$[-3, -5/2[$	-3	$[-3, -5/2[$
11011	-2	$[-5/2, -2[$	-2	$[-5/2, -2[$
11100	-2	$[-2, -3/2[$	-2	$[-2, -3/2[$
11101	-1	$[-3/2, -1[$	-1	$[-3/2, -1[$
11110	-1	$[-1, -1/2[$	-1	$[-1, -1/2[$
11111	0	$[-1/2, 0[$	0	$[-1/2, 0[$

TAB. 1 – Calcul effectué dans la fonction de sélection S pour la base 2 à gauche et 4 à droite.

On pourrait penser qu'utiliser l'ensemble de chiffres $\mathcal{E}_{4,3}$ en base 4 ne présente pas d'intérêt puisqu'il nécessite une fonction de sélection et des petits produits plus complexes. En pratique ce n'est pas vrai. L'ensemble de chiffres utilisé pour une base donnée influence la largeur de l'intervalle dans lequel se trouve l'argument d'entrée x. Plus l'ensemble de chiffres est vaste plus l'intervalle utilisable pour x est grand. Un ensemble comme $\mathcal{E}_{4,3}$ en base 4 permet donc d'avoir un domaine d'utilisation de l'approximation plus grand (modulo le fait que le polynôme ou la fraction rationnelle soit choisi en conséquence). En pratique passer de l'ensemble $\mathcal{E}_{4,2}$ à $\mathcal{E}_{4,3}$ permet de doubler la largeur utilisable du domaine de x. Ceci s'explique par la plus grande latitude de correction possible à chaque itération avec les "grands" chiffres supplémentaires.

L'architecture globale de l'opérateur d'évaluation de fraction rationnelle est présentée en figure 3. L'ar-

chitecture pour la version polynomiale est similaire. Il suffit de remplacer les unités par celles optimisées dans le cas polynomial et de supprimer la mémoire qui stocke les coefficients du polynôme Q.

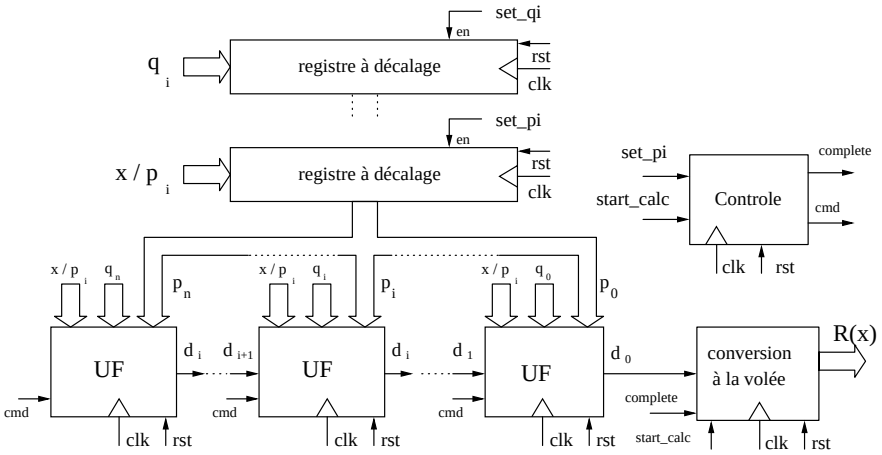


FIG. 3 – Architecture globale de l'opérateur d'évaluation de fraction rationnelle.

Dans cet opérateur, le calcul est fait en parallèle sur les $n+1$ unités pendant m itérations. Les communications pendant le calcul se limitent à la propagation des chiffres $d_i[j]$ d'une unité à la suivante. Au début de chaque nouvelle évaluation, il faut charger la nouvelle valeur de x et réinitialiser les restes partiels avec les coefficients du polynôme P. A chaque changement de fonction à évaluer f, il faut modifier dans les unités les coefficients du polynôme Q.

3.2 Résultats d'implantation des opérateurs

Dans cette section, nous présentons les résultats d'implantation de nos opérateurs en taille et vitesse. Nous avons implanté nos opérateurs sur des FPGA XCV300E Xilinx de la famille Virtex E (avec 3072 slices utilisables au total). Les outils utilisés pour la synthèse et le placement/routage sont les outils propriétaires Xilinx de l'environnement ISE 5.2i (avec XST pour la synthèse). Les résultats sont obtenus avec un effort d'optimisation normal en vitesse.

Deux fonctions cibles sont utilisées pour ces implantations. La fonction exponentielle $\exp(x)$ sur l'intervalle $[0, 1/8]$ et la fonction sinus $\sin(x)$ sur l'intervalle $[0, \pi/32]$. Différentes précisions cibles sont testées pour chaque fonction et chaque type d'architecture (polynomiale ou rationnelle). A chaque fois, les coefficients sont trouvés par l'algorithme `minimax` de Maple. La précision cible est le nombre de bits corrects pour chaque fonction. En pratique nous utilisons une précision intermédiaire de calcul plus grande pour prendre en compte les erreurs lors de l'évaluation numérique du polynôme. Par exemple, pour l'approximation polynomiale de la fonction exponentielle sur 32 bits, nous utilisons une précision interne de calcul (chemin de données et taille des coefficients) de 37 bits. En règle générale, pour un polynôme de degré d, nous ajoutons d bits de précision interne supplémentaires.

La table 2 présente les différents résultats d'implantation pour la base $\beta = 2$ (donc l'ensemble de chiffres $\mathcal{E}_{2,1}$). Les valeurs reportées dans cette table sont : la précision cible (donnée en nombre de bits corrects), le degré du polynôme ou de la fraction rationnelle utilisé, la taille de l'opérateur obtenu (donnée en nombre de slices du FPGA), et la période de l'opérateur obtenu (donnée en nano-seconde).

La table 3 présente les différents résultats d'implantation pour le cas de la base $\beta = 4$ avec l'ensemble de chiffres entre -2 et 2.

La table 4 présente les différents résultats d'implantation pour le cas de la base $\beta = 4$ avec l'ensemble de chiffres entre -3 et 3.

La table 5 présente les différents résultats d'implantation pour le cas de la base $\beta = 8$ avec l'ensemble de chiffres entre -4 et 4.

Nous illustrons sur les courbes en figure 4 le temps de calcul total pour chacune des solutions polynomiales et chacune des solutions rationnelles. Il est clair sur cette figure que la base 8 permet d'obtenir des temps de

Fonction $\exp(x)$, base $\beta = 2$, approximation polynomiale									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	2	3	3	4	4	5	5	6	8
Taille [nb slices]	124	205	253	353	400	529	575	761	1292
Période [ns]	9.92	10.15	9.95	11.78	11.16	12.15	12.53	12.85	14.81

Fonction $\sin(x)$, base $\beta = 2$, approximation polynomiale									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	2	2	3	3	4	5	5	5	7
Taille [nb slices]	124	153	253	297	400	529	575	624	1135
Période [ns]	10.31	11.10	10.54	11.16	11.75	12.32	12.49	12.41	14.50

Fonctions $\exp(x)$ et $\sin(x)$, base $\beta = 2$, approximations rationnelle									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	1	1	2	2	2	3	3	3	4
Taille [nb slices]	153	186	249	295	331	496	538	590	993
Période [ns]	10.77	11.34	12.05	11.70	12.09	13.11	13.15	13.44	15.30

TAB. 2 – Résultats d'implantation pour la base $\beta = 2$.

calcul totaux bien plus faibles. Cette augmentation de la base se paye en termes de surface.

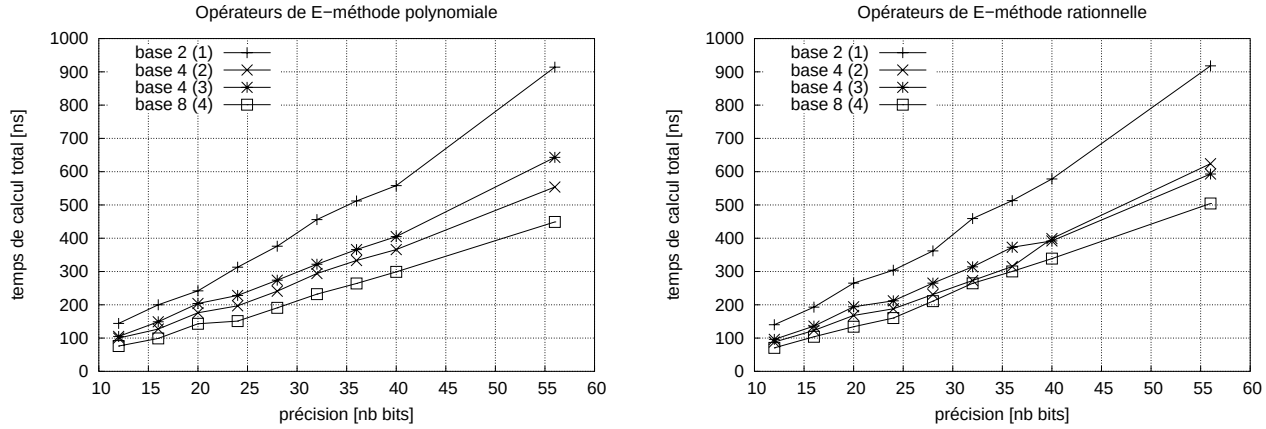


FIG. 4 – Temps de calcul total pour les différentes solutions en fonction de la base (solution polynomiale à gauche et rationnelle à droite).

4 Comparaisons

4.1 Schéma de Horner

Le schéma de Horner permet d'évaluer un polynôme de degré d en utilisant d fois consécutives une brique de base capable de faire un calcul de la forme $XY + Z$ (appelé FMA en arithmétique pour *fused multiply and add*). Dans le cas d'un polynôme de degré 4 on a :

$$P(x) = p_0 + p_1 x + p_2 x^2 + p_3 x^3 + p_4 x^4 = p_0 + \Big(p_1 + \big(p_2 + (p_3 + p_4 x)x\big)\,x\Big)\,x.$$

Nous avons implanté (dans les mêmes conditions que dans la section 3.2) différents opérateurs d'approximation polynomiale utilisant le schéma de Horner. L'opérateur est composé d'un étage FMA, des registres

Fonction $\exp(x)$, base $\beta = 4$ ($\mathcal{E}_{4,2}$), approximation polynomiale									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	2	3	3	4	4	5	5	6	8
Taille [nb slices]	119	191	238	356	412	568	631	796	2082
Période [ns]	14.35	14.63	15.34	14.96	15.04	15.90	16.25	16.42	17.52

Fonction $\sin(x)$, base $\beta = 4$ ($\mathcal{E}_{4,2}$), approximation polynomiale									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	2	2	3	3	4	5	5	5	7
Taille [nb slices]	119	149	238	285	412	568	631	665	1834
Période [ns]	14.35	14.17	15.34	14.62	15.04	15.90	16.25	16.25	17.58

Fonctions $\exp(x)$ et $\sin(x)$, base $\beta = 4$ ($\mathcal{E}_{4,2}$), approximations rationnelle									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	1	1	2	2	2	3	3	3	4
Taille [nb slices]	159	203	273	329	378	589	646	719	1574
Période [ns]	13.60	14.47	15.27	14.46	15.42	15.59	16.16	18.58	20.79

TAB. 3 – Résultats d'implantation pour la base $\beta = 4$ avec l'ensemble de chiffres $\mathcal{E}_{4,2} = \{-2, -1, 0, 1, 2\}$.

pour stocker les coefficients du polynôme (les p_i) et la logique de contrôle de l'opérateur. Les résultats d'implantation sont présenté dans la table 6.

On constate que les résultats des opérateurs à base de schéma de Horner sont nettement moins bons que ceux à base de E-méthode (cas polynomial). Par exemple, pour une précision de 32 pour la fonction exp, il faut utiliser 1092 slices au lieu de 661 (avec $\beta = 4$ et $\mathcal{E}_{4,3}$) et une période de 30 ns au lieu de 17.5 ns. Soit 65% et 70% d'augmentation respectivement en taille et en période. En comparant avec une version rationnelle des opérateurs de E-méthode en grande base les écarts sont encore plus grands.

Maintenant ces résultats ne sont pas transposables à tout type de FPGA. Sur les dernières générations de circuit FPGA, il y a des blocs spécialisés pour effectuer des petites multiplications de façon câblée. Pour ces architectures, il serait intéressant de regarder comment se comparent les opérateurs de E-méthode en grande base par rapport à un schéma de Horner utilisant ces petits multiplieurs câblés.

4.2 Schéma de Horner et division SRT

Pour comparer les performances de nos opérateurs dans le cas des fractions rationnelles, nous avons réalisé des opérateurs utilisant le schéma de Horner et un diviseur SRT rapide (en base 4 particulièrement efficace sur FPGA Xilinx). Le diviseur SRT est généré automatiquement à partir d'un outil développé par les auteurs [11]. Les résultats d'implantation sont donnés dans la table 7.

Dans le cas d'une précision de 56 bits, il a été nécessaire de choisir un FPGA plus gros car le nombre de slices disponibles dans un XCV300E n'était pas suffisant (nous avons utilisé un XCV600E pour ce cas particulier).

Du fait du coût important de la division (même si le générateur divgen donne de très bons diviseurs), les résultats pour les approximations rationnelles avec schéma de Horner et division sont nettement moins bons que ceux avec des opérateurs rationnels de E-Méthode en grande base. Par exemple, pour une précision de 32 bits il faut 1842 slices et une période de 28.4 ns pour le schéma « Horner et division » contre seulement 519 slices et une période de 19.3 ns avec la E-méthode en base 8.

5 Conclusion et perspectives

Ce travail présente l'étude et l'implantation FPGA d'opérateurs arithmétiques, à base d'additions et de décalages, pour l'approximation polynomiale et rationnelle de fonctions en grande base. Ces opérateurs sont

Fonction $\exp(x)$, base $\beta = 4$ ($\mathcal{E}_{4,3}$), approximation polynomiale									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	2	3	3	4	4	5	5	6	8
Taille [nb slices]	133	220	266	402	462	661	727	934	1677
Période [ns]	15.00	16.94	17.74	17.21	17.15	17.43	17.84	17.77	21.78

Fonction $\sin(x)$, base $\beta = 4$ ($\mathcal{E}_{4,3}$), approximation polynomiale									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	2	2	3	3	4	5	5	5	7
Taille [nb slices]	133	161	266	316	462	661	727	801	1469
Période [ns]	15.00	16.55	17.74	16.91	17.15	17.43	17.84	18.02	20.40

Fonctions $\exp(x)$ et $\sin(x)$, base $\beta = 4$ ($\mathcal{E}_{4,3}$), approximations rationnelle									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	1	1	2	2	2	3	3	3	4
Taille [nb slices]	183	229	297	353	406	669	736	816	1474
Période [ns]	14.80	16.05	17.61	16.30	17.67	17.94	19.13	18.23	19.78

TAB. 4 – Résultats d'implantation pour la base $\beta = 4$ avec l'ensemble de chiffres $\mathcal{E}_{4,3} = \{-3, -2, -1, 0, 1, 2, 3\}$.

des versions en grande base de l'itération de E-méthode proposée par M. Ercegovac dans [4, 6]. Jusqu'ici seules des versions en base 2 de cette méthode étaient proposées. Nous montrons que des bases plus grandes comme 4 ou 8 sont très intéressantes d'un point de vue de la vitesse tout en se limitant à des tailles modérées de circuit.

Nous pensons travailler, dans le futur, à la réalisation d'un générateur automatique pour ces opérateurs. Nous souhaitons aussi aborder le cas de cibles ASIC, ce qui va nécessiter de prendre en compte des systèmes redondants de représentation des nombres pour toutes les valeurs intermédiaires (les restes partiels w). Ceci risque de fortement compliquer la fonction de sélection, un travail important devra probablement être fait à ce niveau.

Références

[1] Brisebarre (N.), Hennecart (F.), Muller (J.-M.), Tisserand (A.) et Torres. (S.). – MEPLib. – http ://lipforge.ens-lyon.fr/, 2004.

[2] Brisebarre (N.) et Muller (J.-M.). – Functions approximable by e-fractions. *In : 38th Conference on signals, systems and computers.* – Pacific Grove, California, US, novembre 2004.

[3] de Dinechin (F.) et Tisserand (A.). – Some improvements on multipartite tables methods. *In : 15th International Symposium on Computer Arithmetic ARITH15*, éd. par Burgess (N.) et Ciminiera (L.). pp. 128–135. – Vail, Colorado, juin 2001.

[4] Ercegovac (M. D.). – *A general method for evaluation of functions and computation in a digital computer.* – Thèse de PhD, Dept. of Computer Science, University of Illinois, Urbana-Champaign, 1975.

[5] Ercegovac (M. D.) et Lang (T.). – *Division and Square-Root Algorithms : Digit-Recurrence Algorithms and Implementations.* – Kluwer Academic, 1994.

[6] Ercegovac (M.D.). – A general hardware-oriented method for evaluation of functions and computations in a digital computer. *IEEE Transactions on Computers*, vol. C-26, n7, 1977, pp. 667–680.

[7] Ercegovac (M.D.), Lang (T.), Muller (J.-M.) et Tisserand (A.). – Reciprocation, square root, inverse square root, and some elementary functions using small multipliers. *IEEE Transactions on Computers*, vol. 49, n7, juillet 2000, pp. 627–637.

Fonction $\exp(x)$, base $\beta = 8$ ($\mathcal{E}_{8,4}$), approximation polynomiale									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	2	3	3	4	4	5	5	6	8
Taille [nb slices]	244	408	477	705	801	1075	1386	1583	3070
Période [ns]	16.30	17.89	18.59	17.13	17.86	18.80	19.31	19.85	21.58

Fonction $\exp(x)$, base $\beta = 8$ ($\mathcal{E}_{8,4}$), approximation polynomiale									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	2	2	3	3	4	5	5	5	7
Taille [nb slices]	244	300	477	551	801	1075	1386	1416	2834
Période [ns]	16.30	16.51	18.59	16.82	17.86	18.80	19.31	19.91	21.38

Fonctions $\exp(x)$ et $\sin(x)$, base $\beta = 8$ ($\mathcal{E}_{8,4}$), approximation rationnelle									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	1	1	2	2	2	3	3	3	4
Taille [nb slices]	345	448	585	672	778	1309	1472	1612	2779
Période [ns]	16.29	18.34	18.29	18.47	21.10	22.68	23.09	23.63	25.22

TAB. 5 – Résultats d'implantation pour la base $\beta = 8$ avec l'ensemble de chiffres $\mathcal{E}_{8,4} = \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$.

Fonction $\exp(x)$, approximation polynomiale avec schéma de Horner									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré du polynôme	2	3	3	4	4	5	5	6	8
Taille [nb slices]	175	318	455	642	801	1092	1301	1600	3029
Période [ns]	18.30	21.31	21.15	23.13	24.22	30.06	35.16	29.10	49.34

Fonction $\sin(x)$, approximation polynomiale avec schéma de Horner									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	2	2	3	3	4	5	5	5	7
Taille [nb slices]	175	263	455	591	801	1092	1301	1516	2908
Période [ns]	18.30	22.11	21.15	23.11	24.26	30.06	35.16	29.70	49.09

TAB. 6 – Résultats d'implantation pour le schéma de Horner (approximation polynomiale).

[8] Ercegovac (M.D.), Muller (J.M.) et Tisserand (A.). – FPGA implementation of polynomial evaluation algorithm. *In : Field Programmable Gate Arrays for Fast Board Development and Reconfigurable Computing*, éd. par SPIE, pp. 177–188. – octobre 1995.

[9] Hart (J.F.). – *Computer Approximations*. – Wiley, 1968.

[10] Mencer (O.), Morf (M.), Liddicoat (A.) et Flynn (M. J.). – Efficient digit-serial rational function approximations and digital filtering applications. *In : Asilomar Conference on Signals, Systems, and Computers*, éd. par IEEE. – novembre 1999.

[11] Michard (R.), Tisserand (A.) et Veyrat-Charvillon. (N.). – Divgen : a divider circuit generator. – http ://lipforge.ens-lyon.fr/, 2004.

[12] Muller (J.-M.). – *Elementary Functions : Algorithms and Implementation*. – Birkhäuser, Boston, 1997.

[13] Pineiro (J. A.), Bruguera (J. D.) et Muller (J.-M.). – Faithful powering computation using table look-up and a fused accumulation tree. *In : Proceedings of the 15th IEEE Symposium on Computer Arithmetic*. pp. 40–47. – IEEE Computer Society, 2001.

Fonctions $\exp(x)$ et $\sin(x)$, approximation rationnelle avec schéma de Horner et division									
Précision [nb bits]	12	16	20	24	28	32	36	40	56
Degré	1	1	2	2	2	3	3	3	4
Taille [nb slices]	352	524	786	1033	1348	1842	2222	2643	4869
Période [ns]	18.06	24.83	25.02	25.30	26.00	28.34	30.27	36.31	33.41

TAB. 7 – Fonctions $\exp(x)$ et $\sin(x)$, approximation rationnelle avec schéma de Horner et division SRT.

[14] Remes (E.). – Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation. *C.R. Acad. Sci. Paris*, vol. 198, 1934, pp. 2063–2065.

[15] Schulte (M.) et Stine (J.). – Approximating elementary functions with symmetric bipartite tables. *IEEE Transactions on Computers*, vol. 48, n8, août 1999, pp. 842–847.

[16] Volder (J.). – The CORDIC computing technique. *IRE Transactions on Computers*, vol. EC-8, n3, 1959, pp. 330–334.

[17] Vuillemin (Jean). – Exact real computer arithmetic with continued fractions. *In : Proc. of the 1988 ACM conference on LISP and functional programming (LFP'88)*. pp. 14–27. – ACM Press, 1988.

[18] Walther (J.). – A unified algorithm for elementary functions. *In : Joint Computer Conference Proceedings*. – 1971. Reprinted in E. E. Swartzlander, *Computer Arithmetic*, Vol. 1, IEEE Computer Society Press Tutorial, Los Alamitos, CA,1990.